(19) Europäisches Patentamt
European Patent Office
Office européen des brevets

(11) EP 0 674 273 B1

(12) EUROPEAN PATENT SPECIFICATION

(45) Date of publication and mention of the grant of the patent:
24.05.2000 Bulletin 2000/21

(51) Int. Cl.[7]: **G06F 13/36**

(21) Application number: **94114617.7**

(22) Date of filing: **16.09.1994**

(54) Atomic operation control scheme

Steuerungsschema von atomaren Operationen

Schéma de commande des opérations atomiques

(84) Designated Contracting States:
DE FR GB

(30) Priority: **24.03.1994 US 217687**

(43) Date of publication of application:
27.09.1995 Bulletin 1995/39

(73) Proprietor:
Hewlett-Packard Company
Palo Alto, California 94304 (US)

(72) Inventors:
- **Bryg, William R.**
  Saratoga, California 95070 (US)
- **Frink, Craig R.**
  Chelmsford, MA 01824 (US)
- **McMahan, Larry N.**
  Fremont, CA 94538 (US)
- **Nusbaum, Helen**
  Sacramento, CA 95818 (US)

(74) Representative:
Schoppe, Fritz, Dipl.-Ing.
Schoppe, Zimmermann & Stöckeler
Patentanwälte
Postfach 71 08 67
81458 München (DE)

(56) References cited:
EP-A- 0 366 432
US-A- 4 975 870
US-A- 5 293 496

EP 0 674 273 B1

Description

[0001] The present invention relates to buses, and more particularly to control of shared buses capable of carrying information among memory, input/output devices, and other components of a computer system.

[0002] Computer systems commonly have a plurality of components, such as processors, memory, and input/output devices, and a shared bus for transferring information among the components. Typically, the components are coupled to the bus in the form of component modules, each of which may contain one or more processors, memory, and/or input/output devices. Input/output modules often consist of input/output adapters that provide an interface between the main system bus and an input/output bus to which one or more input/output devices are coupled.

[0003] Information is transmitted on the system bus among component modules during bus "cycles," each bus cycle being a period of time during which a module has control of the bus and is permitted to transfer a limited quantity of information on the bus. Modules commonly send "transactions," such as conventional "read" and "write" transactions, on the bus to other modules. Each transaction generally takes one or more cycles to complete. The module entitled to control the bus during a given cycle generally is determined by an arbitration among modules that takes place during one or more earlier cycles.

[0004] In many cases, modules are required to perform a set of related transactions consecutively, without data being altered by other transactions while that set of transactions is in progress. In other words, a module may need to deal with a consistent memory image for purposes of certain sets of transactions. Such sets of transactions are generally referred to as "atomic operations" or "atomically-linked transactions." Atomic operations may be necessary, for example, where two or more modules are each required to read the data associated with a particular memory address, operate on the data, and then write the altered data back to that memory address. Software will dictate the order in which these sets of transactions must be performed and incorrect results may be obtained if the sets of transactions are not performed as atomic operations in the required order.

[0005] For example, a processor module may be required to perform an atomic operation consisting of reading data associated with a memory address (which happens to have a value of "X"), clearing the data, and writing the results of its operations (*i.e.*, "0") to the same address. An input/output module may be required to subsequently perform an atomic operation consisting of reading data (now having a value of "0") from the same memory address, incrementing the data by one, and then writing the results (*i.e.*, "1") to the same address. Thus, the results of these two consecutive atomic operations should be that a value of "1" is stored at the relevant memory address.

[0006] In order to ensure correct results, however, it is necessary to make certain that the processor module has completed its atomic operation before the input/output module reads the data. If the input/output module were to read the data before the processor module has completed its write transaction, incorrect results will be obtained regardless of the order in which the write transactions are performed. Specifically, the input/output module would read a value of "X," increment the data, and subsequently write a value of "X + 1" to that memory address. Assuming that the processor module had already completed its write transaction by the time the input/output module writes this value, the incorrect value "X + 1" would remain at the relevant memory address after each module had performed its operations. If the processor module had not yet completed its write transaction, it would subsequently write an incorrect value of "0" to the address.

[0007] In many conventional bus systems, proper execution of atomic operations is assured by allowing a module to "lock" the bus, that is to prevent any other module from using the bus, while that module is performing atomic operations. Thus, if a module must perform an atomic operation, it simply locks the bus for the number of cycles necessary to complete all atomically-linked transactions and subsequently releases its "lock," allowing other modules to utilize the bus.

[0008] If multiple buses are linked and the transaction involves transmitting information over more than one bus, it is generally necessary to lock all buses while the atomically-linked transactions are being performed. For example, conventional EISA ("Extended Industry Standard Architecture") cards for transmitting information on an EISA bus are capable of issuing up to 64 atomically-linked transactions. The EISA bus may be coupled through a bus adapter to an input/output bus, which is in turn coupled through an input/output bus adapter to a main computer bus that is coupled to a computer's main memory. Thus, the cards may interface with the computer's main memory through an EISA bus, an input/output bus, and a main computer bus. In many prior art systems, the input/output bus and the main computer bus are both locked during the entire period necessary for the 64 atomically-linked transactions, and no other module is entitled to win arbitration for the computer bus during this period. A disadvantage of locking the bus is that many bus cycles may be "wasted" during the period in which the bus is locked. The reason for this is that each atomically-linked transaction may involve processing time during which time the bus is not used. Additionally, the latency time of the memory during read transactions may be several bus cycles.

[0009] Processing atomic operations is somewhat more complicated in the case of "split transaction" buses, especially where one or more modules coupled to the bus has a cache memory. Split transaction buses are designed to reduce the impact of delays associated with memory latency by allowing modules to issue transactions while earlier

issued transactions are in progress. For example, in split transaction buses, a response to a read transaction need not immediately follow a request, and the bus may be used for other transactions during the period in which the requested data is being retrieved. When the responding module is ready to return the requested data, the responding module arbitrates to obtain control of the bus and then sends the requested data to the requesting module. Thus, split transaction buses generally eliminate the need to "waste" bus cycles while requests for data are being processed.

[0010] Locking a split transaction bus can result in the system becoming deadlocked. Specifically, a module may lock the bus while one or more split transactions previously issued by other modules are in progress (that is, have not been responded to). One of the atomically-linked transactions may require data that cannot be obtained until an earlier issued transaction is completed, but the earlier issued transaction cannot be completed because the bus is locked.

[0011] As an example of a system becoming deadlocked in the above manner, it is useful to consider the case of a shared-memory multiprocessor computer system in which one or more processors has a cache memory. In such a computer, the most current data associated with a particular memory address at any given time may be stored in one or more cache memories, and/or in the main memory. When a module requests data from a memory address, a conventional "cache coherency scheme" ensures that the most current data is supplied by the module or memory having the current data.

[0012] Assuming an input/output module locks the bus for an atomic operation, however, one of the atomically-linked transactions may request data whose current value is stored only in a processor's cache memory. Since the bus is locked, the processor is unable to arbitrate for the bus to return the data, and the system becomes deadlocked. It might be possible to detect the deadlock and provide for temporary removal of the lock, but such a design would not meet EISA standards for atomicity. Furthermore, the added hardware would increase the cost of the system.

[0013] Accordingly, there is a need for a means for accommodating atomic operations on a split transaction bus without causing deadlock.

[0014] US-A-5,293,496 describes a computer architecture having a time-shared bus which is used by multiple users including private cache systems. During a cache cycle, a bus lockout is prevented by issuing an inhibit write instruction to the bus inhibiting other modules issuing a write-to-memory instruction, whereas bus requests other than a write request may be issued to the bus during the state in which the inhibit write signal is active.

[0015] EPO 366432A2 relates to a method for a bus lock during atomic computer operations. The computer system has a plurality of processors sharing common memory and data bus structures and is operable to perform atomic operations. Once an atomic operation is issued, the processor performing this operation prevents memory access interruptions by other processors by locking out same during the atomic operation. To avoid a system deadlock, a lock time out is used so that after a timer expired, the lock is automatically released.

[0016] It is the object of the present invention to provide an improved split transaction bus system accommodating atomic operations on the bus without causing a deadlock and without unnecessary delays due to locking the bus.

[0017] This object is achieved by a bus system according to claim 1.

[0018] The present invention is a bus system for coupling a set of component modules to a memory. Each component module is capable of issuing and receiving transactions on a bus. The transactions on the bus are divided into two or more types, including a first transaction type that alters or samples the image of memory relative to the image of memory at the time a transaction of the first transaction type is placed on the bus and a second transaction type that does not alter or sample the memory image existing at the time a transaction of the second transaction type is placed on the bus.

[0019] The bus system includes a bus coupling the component modules to each other and to the memory. Each component module includes means responsive to an external signal for limiting the type of transactions issued by the module on the bus. At least one module also includes means for generating an atomic signal indicative of the module's need to issue one or more atomic transactions.

[0020] The bus system also includes a bus control means, responsive to the atomic signal, for generating a transaction type allowed signal to the component modules. The transaction type allowed signal prevents the component modules from issuing transactions of the first transaction type.

FIG. 1 is a block diagram illustrating the preferred embodiment of a bus system utilizing an atomic operation control scheme according to the present invention.

[0021] The present invention encompasses a split transaction bus that may be used with a computer system and accommodates atomic operations without locking the bus and without the possibility of deadlock during the atomic operations. The computer system includes a bus, component modules that send transactions to each other on the bus, and a bus controller that specifies the types of transactions that can be sent on the bus at any given time.

[0022] The bus controller is preferably connected to the main memory used by the component modules to store information. One or more of the modules may also have local cache memories. The contents of memory at any given time may be viewed as a list of addresses and data values associated with each address. In general, there is one loca-

tion in main memory associated with each memory address in some predetermined range of addresses. The location of the most current data associated with one of these addresses may be in main memory or in one of the cache memories located in some of the component modules. A memory image at time t is functionally equivalent to a list of the addresses in the above-mentioned range and the data associated with each address at the time t. For convenience, time t will be defined as the time at which a transaction is placed on the bus.

[0023] Two classes of transactions can be identified with respect to time t, those that change the memory image from its value at time t or sample the memory image, and those that do not change or sample the memory image. An input/output write transaction issued after time t changes the memory image, since it alters the data associated with at least one address. A read transaction samples the memory image. In contrast, the return of data to a processor in response to a transaction issued before time t or a cache-to-cache copy do not alter or sample the memory image. The data return provides data associated with the image at a time before time t. The cache-to-cache copy operation does not change any data value since such a transaction merely changes the physical location at which the data associated with an address is stored. Similarly, the write-back of private/dirty data from a processor's cache to memory does not change the data value, since the transaction merely changes the physical location of the data from a cache to the main memory.

[0024] When one module is performing an atomic operation, the bus controller limits transactions to those that do not change the memory image at the time the atomic operation is commenced. The bus controller, however, permits responses or returns of data, assuming the response or return does not alter the memory image.

[0025] A block diagram of an exemplary computer system according to the present invention is shown at 10 in FIG. 1. Computer system 10 is a multiprocessor computer having a bus 12 and a plurality of components coupled to bus 12. The components include a main memory controller 14, input/output modules 16 and 18, and processor modules 20 and 22. The components communicate with one another by sending and receiving transactions on bus 12. The components may arbitrate for control of bus 12 using any conventional arbitration scheme.

[0026] Bus 12 is a high performance processor-memory-I/O interconnect bus. Bus 12 is a split transaction bus. For example, after a READ transaction is issued on bus 12, the module that issued the READ relinquishes the bus allowing other modules to use the bus for other transactions. When the requested data is available, the responding module for the READ arbitrates for the bus, and then transmits the data. WRITE transactions are not split, so the master transmits the WRITE data immediately following the address cycle.

[0027] Processor modules 20 and 22 are the main processors for computer system 10, and software for the system executes simultaneously on all processors.

[0028] Input/output modules 16 and 18 serve as interfaces between computer system 10 and input/output devices. Input/output modules 16 and 18 each contain at least one input/output adaptor that is coupled between bus 12 and an input/output device, generally through an input/output bus. An exemplary input/output bus 30 and an exemplary input/output device 32 are shown coupled to input/output module 18.

[0029] Input/output modules 16 and 18 control STOP_MOST lines 26 and 28, respectively. The STOP_MOST lines allow the input/output modules to take effective control of the memory system for atomic operations. This is useful, for example, when several memory operations must be performed by a module without other modules reading data from or writing data to relevant memory addresses. When an input/output module asserts STOP_MOST, it becomes "atomic owner" of bus 12. As explained further below, when STOP_MOST is asserted, only the atomic owner is allowed to issue new transactions. Other modules are allowed to issue only return or response-type transactions that do not change the current memory image. If more than one input/output module simultaneously asserts STOP_MOST, any conventional arbitration algorithm may be used to determine which input/output module becomes atomic owner.

[0030] Main memory controller 14 is responsible for reading information from the main memory 15 and storing information in main memory 15 in a conventional manner. Main memory controller 14 preferably also serves as the "host" module or "bus controller" for purposes of dictating the manner in which bus 12 may be used by the remaining modules, which can be considered client modules. Specifically, main memory controller 14 controls a CLIENT_OP line 24, which is coupled directly to each client module. Main memory controller 14 sends signals to each client module on CLIENT_OP line 24 to indicate what types of transactions may be placed on bus 12 during the next available cycle. CLIENT_OP line 24 is used to limit the types of transactions issued on the bus when STOP_MOST is being asserted for an atomic operation. The use of CLIENT_OP line 24, along with its use for accommodating atomic operations, is explained in more detail below.

[0031] In addition to the main memory, any or all client modules (both processor and input/output) may have a conventional cache memory for storing recently used data. Ordinarily, a cache memory stores both the frequently used data and the addresses where these data items are stored in main memory. When the processor seeks data from an address in memory, it requests that data from the cache memory using the address associated with the data. The cache memory checks to see whether it holds data associated with that address. If so, the cache memory returns the requested data directly to the processor. If the cache memory does not contain the desired information (i.e., a "cache miss" occurs), the cache requests the data from main memory and stalls the processor while it is waiting for the data.

Since cache memory is faster than main RAM memory, this strategy results in improved system performance.

[0032] In the case of a shared memory multi-processor computer such as computer system 10, the situation is somewhat more complex. The most current data may be stored in one or more cache memories, or in the main memory. Software executing on the processors must utilize the most current values for data associated with a particular address. Thus, a "cache coherency scheme," must be implemented to assure that all copies of data for a particular address are the same. Many conventional cache coherency schemes are available and are widely known in the art.

[0033] In a typical write-back coherency scheme, when data is requested by a module, each module having cache memory performs a "coherency check" of its cache memory to determine whether it has data associated with the requested address and reports the results of its coherency check. Each module also generally reports the status of the data stored in its cache memory in relation to the data associated with the same address stored in main memory and other cache memories. For example, a module may report that its data is "private" (i.e., the data value is only usable by this module) or that the data is "shared" (i.e., the data may reside in more than one cache memory at the same time). A module may also report whether its data is "clean" (i.e., the same as the data associated with the same address stored in main memory) or "dirty" (i.e., the data has been changed after it was obtained).

[0034] The results of the coherency checks performed by each module are analyzed by a selected processor and the most current data is provided to the module that requested the data. A "coherent transaction" is any transaction that requires a check of other caches to see whether data associated with a memory address is stored in the other caches, or to verify that data is current. Most reads and some writes to memory are coherent transactions. Those skilled in the art are familiar with many types of coherent transactions, such as a conventional read private, and non-coherent transactions, such as a conventional write-back.

[0035] In a preferred embodiment of the present invention, the CLIENT_OP bus supports the signals shown in Table 1.

TABLE 1

Name	Value	Significance for Next Available Cycle
NONE_ALLOWED	010	No TRANS_ALLOWED, but clients still control bus
RET_ONLY	100	Return or response TRANS_ALLOWED
ATOMIC	110	Client who is "atomic owner" can issue any transaction, other clients can issue only responses.
ANY_TRANS	111	Any transaction allowed

[0036] The ATOMIC client option signal relates directly to atomic control of the bus. The other CLIENT_OP signals from Table 1 are not critical to atomic control of the bus according to the present invention. These other signals are included, for illustrative purposes, to show other possible ways that the CLIENT_OP bus may limit the transactions allowed during a given cycle if the ATOMIC client option signal is not in use. These other CLIENT_OP signals will be briefly explained as well.

[0037] A client option signal of ANY_TRANS indicates that any transaction is allowed during the next available cycle.

[0038] The "ATOMIC" client option signal is generated in direct response to a client asserting STOP_MOST. The ATOMIC client option signal allows the client asserting STOP_MOST to perform several atomically-linked transactions on bus 12. When ATOMIC is asserted, all other clients are only allowed to respond to earlier sent transactions, or to send data in a manner that does not alter the memory image, such as write-backs of previously held private-dirty cache lines. The following types of conventional transactions are preferably allowed while a client option signal of ATOMIC is asserted: cache-to-cache writes, write-backs, error notification transactions, input/output data returns, memory data returns.

[0039] The above-listed transactions are allowed during an ATOMIC client option signal because they do not change the image of memory viewed by the input/output device performing the atomic operation. In other words, the most current data associated with an memory address will not be altered by these transactions. The allowed transactions may, however, move data from one location in the computer system (*e.g.,* a cache memory) to another location (*e.g.,* a different cache memory or the main memory). Moving the data in this manner, however, will not affect the most current data value associated with each memory address and, therefore, the input/output device will deal with a consistent memory image whether or not any allowed transactions are performed during the ATOMIC client option signal.

[0040] New reads are not allowed by other modules while ATOMIC is asserted because they would result in the memory being sampled in the midst of an atomic operation. This may result in an incorrect data value being read if sub-

sequent atomically-linked transactions within the relevant atomic operation alter the data that was read.

[0041] The RET_ONLY is an optional, exemplary client option signal that indicates that only returns (write-backs) of previously held private-dirty cache lines, or responses to previous transactions are allowed. For example, if processor module 20 issues a read of data that is held private-dirty in processor 22's cache, processor 22 can supply that cache line in a conventional cache-to-cache copy. That cache-to-cache copy transaction can be initiated under the influence of a RET_ONLY client option signal, since the cache-to-cache copy is generally a response to a read and does not require a module to accept a new transaction for processing. Similarly, input/output module 16 can return data from an earlier input/output read transaction under the influence of a RET_ONLY client option signal.

[0042] It will be appreciated that, when there is an atomic owner, the effective client option signal for the atomic owner is ANY_TRANS and the effective client option signal for all other clients is RET_ONLY.

[0043] In accordance with the present invention, an input/output module that needs to perform an atomic operation will assert its STOP_MOST line, which is coupled to main memory controller 14 and to the other input/output module. In response, main memory controller 14 will change the client option signal to ATOMIC. If more than one input/output module has asserted STOP_MOST, the modules will arbitrate among themselves (according to any conventional arbitrations scheme) to determine which module will perform its atomic operation during the ATOMIC client option signal.

[0044] As explained above, when ATOMIC is asserted, no other modules are permitted to perform transactions that alter the state of memory: that is, no transactions are allowed that change the value of data that is current at the time the atomic operation is commenced. Thus, the atomic operation can be performed effectively without interference from other modules. When the atomic operation is completed, the module de-asserts its STOP_MOST line.

[0045] It will be appreciated by those skilled in the art that memory controller 14 serves as the host, or bus controller, for convenience only, and that host functions and memory control functions may be separated into two or more modules.

[0046] The terms "bus(es)" and "line(s)" have both been used in this detailed description to denote various sets of one or more electrical paths that are more fully described above. It will be appreciated by those skilled in the art that the terms "bus" and "line" are not intended to be mutually exclusive or otherwise limiting in themselves. For example, the terms "CLIENT_OP bus" and "CLIENT_OP lines" have been used interchangeably to denote a set of hardware lines driven only by the host, as described more fully above.

Claims

1. A bus system (10) for coupling a plurality of component modules (16, 18, 20, 22) to a memory (15), each of said component modules being capable of issuing and receiving transactions on a bus (12), comprising:

   a bus (12) coupling said component modules to each other and to said memory (15), one of said component modules including means responsive to a signal external to said one of said component modules for limiting the type of transactions issued by said module on said bus, at least one of said component modules further comprising means for generating an atomic signal indicative of said module's need to issue one or more atomic transactions; and

   bus control means (14), responsive to said atomic signal, and connected to all component modules; characterized in

   that said transactions issued and received by said component modules are divided into a plurality of types, said types including a first transaction type that changes the contents of said memory relative to the contents of said memory at the time a transaction of said first transaction type is placed on said bus (12) or samples the contents of said memory, and a second transaction type that does not change the contents of said memory relative to the contents of said memory at the time a transaction of said second transaction type is placed on said bus (12) or does not sample the contents of said memory;

   said bus control means (14), in response to said atomic signal, generates and couples a transaction type allowed signal to all of said component modules, said transaction type allowed signal preventing one of said component modules from issuing transactions of said first transaction type, while allowing one of said component modules to issue transactions of said second transaction type.

2. The bus system (10) of Claim 1, wherein said modules have means for arbitrating among themselves for control of the bus based on said transaction type allowed signal.

3. The bus system (10) of Claim 1, wherein two of said modules include cache memories and said second transaction

type includes cache-to-cache copies between said two modules.

4. The bus system (10) of Claim 1, wherein said second transaction type includes transactions returning data in response to a transaction issued before said transaction type allowed signal was coupled to said component modules.

5. The bus system (10) of Claim 1, wherein said first transaction type includes transactions that read data from the memory.

6. The bus system (10) of Claim 1, wherein at least one module includes a cache memory and said second transaction type includes write-backs of dirty data from said cache memory to said memory.

Patentansprüche

1. Ein Bussystem (10) zum Koppeln einer Mehrzahl von Komponentenmodulen (16, 18, 20, 22) mit einem Speicher (15), wobei jedes der Komponentenmodule in der Lage ist, Transaktionen auf einem Bus (12) auszugeben und zu empfangen, mit folgenden Merkmalen:

einem Bus (12), der die Komponentenmodule miteinander und mit dem Speicher (15) koppelt, wobei eines der Komponentenmodule eine Einrichtung aufweist, die auf ein Signal von außerhalb des einen der Komponentenmodule anspricht, um den Typ von Transaktionen, die durch das Modul auf den Bus ausgegeben werden, zu begrenzen, wobei zumindest eines der Komponentenmodule ferner eine Einrichtung zum Erzeugen eines atomaren Signals, das den Bedarf des Moduls, eine oder mehrere atomare Transaktionen auszugeben, anzeigt, aufweist;

einer Bus-Steuereinrichtung (14), die auf das atomare Signal anspricht und mit allen Komponentenmodulen verbunden ist;
dadurch gekennzeichnet,

daß die Transaktionen, die durch die Komponentenmodule ausgegeben und empfangen werden, in eine Mehrzahl von Typen unterteilt sind, wobei die Typen einen ersten Transaktionstyp, der den Inhalt des Speichers relativ zu dem Inhalt des Speichers zu dem Zeitpunkt, zu dem eine Transaktion des ersten Transaktionstyps auf dem Bus (12) plaziert wird, ändert oder den Inhalt des Speichers abtastet, und einen zweiten Transaktionstyp umfassen, der den Inhalt des Speichers relativ zu dem Inhalt des Speichers zu dem Zeitpunkt, zu dem eine Transaktion des zweiten Transaktionstyps auf dem Bus (12) plaziert wird, nicht ändert oder den Inhalt des Speichers nicht abtastet;

daß die Bussteuereinrichtung (14) ansprechend auf das atomare Signal ein Transaktionstyp-Erlaubt-Signal erzeugt und zu allen Komponentenmodulen koppelt, wobei das Transaktionstyp-Erlaubt-Signal verhindert, daß eines der Komponentenmodule Transaktionen des ersten Transaktionstyps ausgibt, während es erlaubt, das eines der Komponentenmodule Transaktionen des zweiten Transaktionstyps ausgibt.

2. Das Bussystem (10) nach Anspruch 1, bei dem die Module eine Einrichtung aufweisen, um untereinander eine Entscheidung für die Steuerung des Busses basierend auf dem Transaktionstyp-Erlaubt-Signal durchzuführen.

3. Das Bussystem (10) nach Anspruch 1, bei dem zwei der Module Cache-Speicher umfassen, wobei der zweite Transaktionstyp Cache-Zu-Cache-Kopien zwischen den zwei Modulen aufweist.

4. Das Bussystem (10) nach Anspruch 1, bei dem der zweite Transaktionstyp Transaktionen umfaßt, die Daten ansprechend auf eine Transaktion, die ausgegeben wurde, bevor das Transaktionstyp-Erlaubt-Signal zu den Komponentenmodulen gekoppelt wurde, zurückgeben.

5. Das Bussystem (10) nach Anspruch 1, bei dem der erste Transaktionstyp Transaktionen umfaßt, die Daten aus dem Speicher lesen.

6. Das Bussystem (10) nach Anspruch 1, bei dem zumindest ein Modul einen Cache-Speicher aufweist, wobei der zweite Transaktionstyp Zurückschreibeaktionen von schmutzigen Daten aus dem Cache-Speicher in den Speicher aufweist.

Revendications

1. Système de bus (10) pour le couplage d'une pluralité de modules de composant (16, 18, 20, 22) avec une mémoire (15), chacun desdits modules de composant pouvant délivrer et recevoir des transactions sur un bus (12), comprenant :

   - ◇ un bus (12) couplant lesdits modules de composant l'un à l'autre et à ladite mémoire (15), un desdits modules de composant comprenant un moyen sensible à un signal externe audit module desdits modules de composant pour limiter le type de transactions délivrées par ledit module sur ledit bus, au moins un desdits modules de composant comprenant, de plus, un moyen pour générer un signal atomique indicatif du besoin dudit module de délivrer une ou plusieurs transactions atomiques ; et
   - ◇ un moyen de commande de bus (14), sensible audit signal atomique, et connecté à tous les modules de composant ;

   caractérisé en ce que :

   - lesdites transactions délivrées et reçues par lesdits modules de composant sont divisées en une pluralité de types, lesdits types comprenant un premier type de transaction modifiant le contenu de ladite mémoire par rapport au contenu de ladite mémoire au moment où une transaction dudit premier type de transaction est placée sur ledit bus (12) ou échantillonne le contenu de ladite mémoire, et un second type de transaction ne modifiant pas le contenu de ladite mémoire par rapport au contenu de ladite mémoire au moment où une transaction dudit second type de transaction est placée sur ledit bus (12) et n'échantillonne pas le contenu de ladite mémoire ;
   - ledit moyen de commande de bus (14), en réponse audit signal atomique, génère et couple un signal autorisé de type de transaction vers tous lesdits modules de composant, ledit signal autorisé de type de transaction empêchant un desdits modules de composant de délivrer des transactions dudit premier type de transaction, tout en autorisant un desdits modules de composant de délivrer des transactions dudit second type de transaction.

2. Système de bus (10) selon la revendication 1, dans lequel lesdits modules ont un moyen pour un arbitrage mutuel pour la commande du bus sur la base dudit signal autorisé de type de transaction.

3. Système de bus (10) selon la revendication 1, dans lequel deux desdits modules comprennent des mémoires de cache et ledit second type de transaction comprend des copies de cache à cache entre lesdits deux modules.

4. Système de bus (10) selon la revendication 1, dans lequel ledit second type de transaction comprend des transactions renvoyant des données en réponse à une transaction délivrée avant que ledit signal autorisé de type de transaction ait été couplé auxdits modules de composant.

5. Système de bus (10) selon la revendication 1, dans lequel ledit premier type de transaction comprend des transactions lisant des données à partir de la mémoire.

6. Système de bus (10) selon la revendication 1, dans lequel au moins un module comprend une mémoire de cache et ledit second type de transaction comprend des réécritures de données erronées à partir de ladite mémoire de cache vers ladite mémoire.

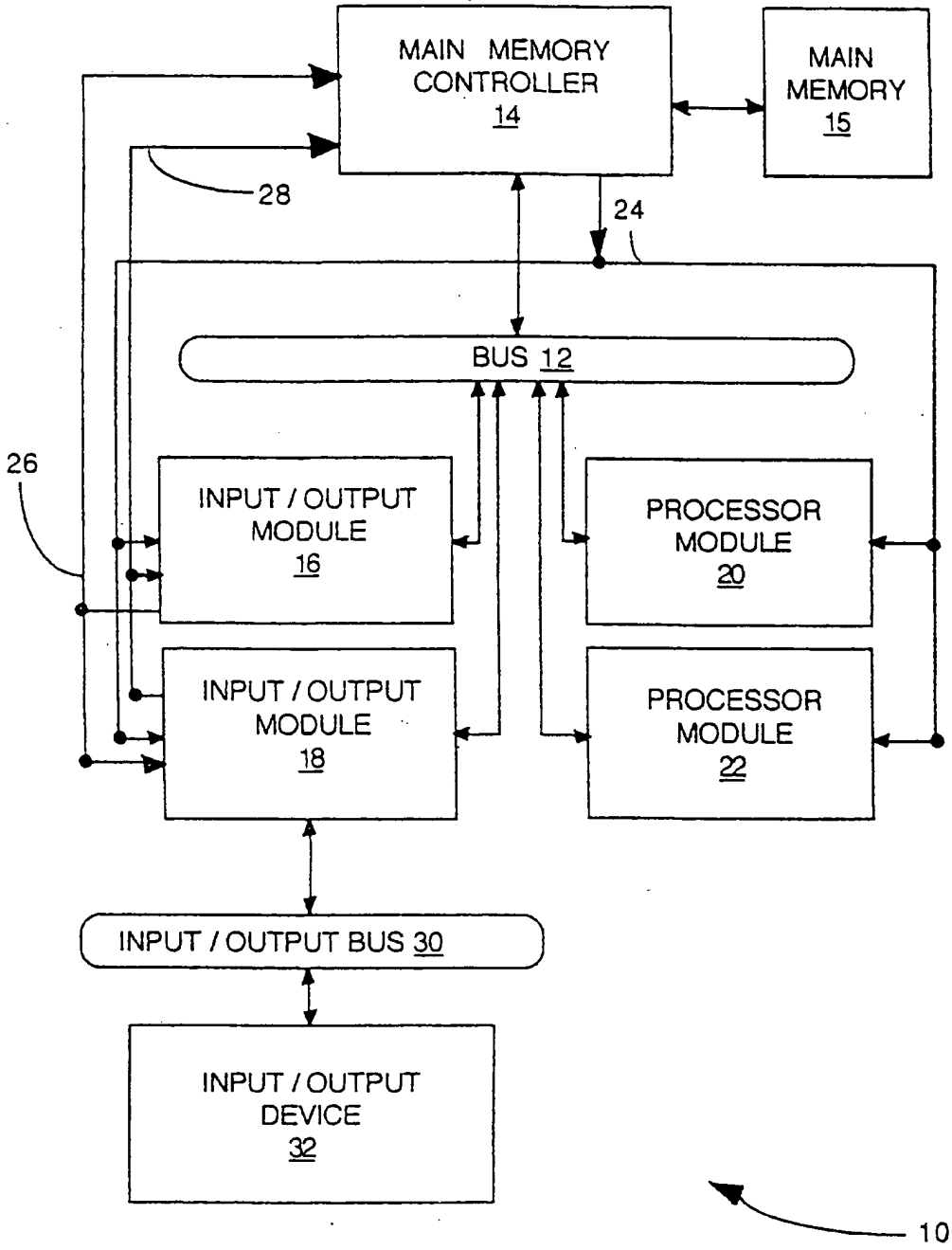

FIG._1